

Comparison of Direct Particle Simulation on the MasPar MP-2 and the Connection Machine CM-2

Leonardo Dagum¹ and Jeffrey D. McDonald²

Report RNR-92-030, December 2, 1992

Submitted to AIAA 28th Thermophysics Conference, Orlando, FL, July 6-9, 1993.

NASA Ames Research Center
Moffett Field, CA 94035

December 2, 1992

Abstract

Particle simulation as applied in the direct simulation Monte Carlo (DSMC) method is a technique for analyzing low density flows and is used extensively for engineering analysis of aerospace vehicles. This work compares two implementations of this method on outwardly similar massively parallel architectures. The MasPar MP-2 and the Connection Machine CM-2 are both massively parallel SIMD architectures with a data parallel paradigm. Internally, however, the two machines differ substantially. The more recent, and more balanced architectural approach of the MasPar MP-2 results in better performance of the MP-2 for direct particle simulation.

¹The author is an employee of Computer Sciences Corporation, M/S T045-1, NASA Ames Research Center.

²The author is an employee of MasPar Computer Corp., 749 N. Mary Ave., Sunnyvale, CA 94086.

COMPARISON OF DIRECT PARTICLE SIMULATION ON THE MASPAR MP-2 AND THE CONNECTION MACHINE CM-2

Leonardo Dagum ¹	Jeffrey D. McDonald
Computer Sciences Corporation	MasPar Computer Corp.
NASA Ames Research Center	749 North Mary Avenue
Moffett Field, CA 94035	Sunnyvale, CA 94086

Abstract

Particle simulation as applied in the direct simulation Monte Carlo (DSMC) method is a technique for analyzing low density flows and is used extensively for engineering analysis of aerospace vehicles. This work compares two implementations of this method on outwardly similar massively parallel architectures. The MasPar MP-2 and the Connection Machine CM-2 are both massively parallel SIMD architectures with a data parallel paradigm. Internally, however, the two machines differ substantially. The more recent, and more balanced architectural approach of the MasPar MP-2 results in better performance of the MP-2 for direct particle simulation.

Introduction

Particle methods are of interest primarily for high altitude, low density flows. When a gas becomes sufficiently rarefied the constitutive relations of the Navier-Stokes equations (i.e. the Stokes law for viscosity and the Fourier law for heat conduction) no longer apply and either higher order relations must be employed (e.g. the Burnett equations [8]), or the continuum approach must be abandoned and the molecular nature of the gas must be addressed explicitly. The latter approach leads to the direct simulation Monte Carlo (DSMC) method pioneered by Bird [1]. Flow conditions requiring particle simulation are typically encountered by flight vehicles operating in the upper atmosphere.

In DSMC, a gas is described by a collection of simulated molecules thus completely avoiding any need for differential equations explicitly describing the flow. By accurately modeling the microscopic state of the gas, the macroscopic description is obtained through the appropriate integration. The primary disadvantage of this approach is that the computational cost is relatively large. Therefore, although the molecular description of a gas is accurate at all densities, a DSMC is competitive only for low densities, where continuum descriptions are not valid. Since the data parallel algorithms for implementing this method have already been developed [4, 5], it is a worthwhile venture to use this method to compare and highlight the architectural differences between the MasPar MP-2 and the Connection Machine CM-2. The current trend towards massively parallel architectures makes this investigation both appropriate and timely.

DSMC encapsulates two grains of parallelism. There is a microscopic or fine-grain parallelism associated with the particles in the simulation and there is a macroscopic or coarse-grain parallelism associated with the cells used in the simulation. Any parallel implementation of a particle simulation needs to address both grains of parallelism.

The natural decomposition using a data parallel programming paradigm is a *data objects* decomposition

¹This work was funded under NASA contract NAS 2-12961

wherein the problem is decomposed in terms of the smallest definable data objects. In the case of particle simulation, one maps the microscopic scale, or more specifically particle data, to individual virtual processors in the machine. The mechanism for obtaining information at the macroscopic or cell scale then requires a sorting of the particle data in cell order on every time step.

Target Architectures

Connection Machine CM-2 Architecture

The Thinking Machines Connection Machine Model CM-2 is a massively parallel Single-Instruction, Multiple-Datastream (SIMD) computer consisting of many thousands of bit-serial data processors under the direction of a front-end computer. The system at NASA Ames Research Center consists of 32K bit-serial processors each with 1 Mbit of memory and operating at 7 MHz. The processors and memory are packaged as 16 in a chip. Each chip also contains the routing circuitry which allows any processor to send and receive messages from any other processor in the system.

In addition, there are 1024 64-bit Weitek floating point processors which are fed from the bit serial processors which yields a peak aggregate performance of 14 GFLOPS. In practice the memory system is incapable of supplying the floating point units with operands at this rate, and the observed performance of the machine even with no interprocessor communication is significantly lower than 14 GFLOPS.

The architecture allows interprocessor communication to proceed in three manners. For very general communication with no regular pattern, the router determines the destination of messages at run time and directs the messages accordingly. This is referred to as general router communication, and has the poorest performance. For communication with an irregular but static pattern, the message paths may be pre-compiled and the router will direct messages according to the pre-compiled paths. This is referred to as compiled communication and is significantly faster than general router communication. Although useful for many applications (e.g. in sparse matrix type calculations), this type of communication is not usable in particle simulation. Finally, for communication which is perfectly regular and involves only shifts along grid axes, the system software optimizes the data layout by ensuring strictly nearest neighbor communication and uses its own pre-compiled paths. The grid communication performance of the CM-2 has been thoroughly investigated in [6, 7]. On a 32k processor CM-2 one measures an overall communication bandwidth of 67 MB/sec using the router and a bandwidth of 450 MB/sec for regular communication [4, 6].

MasPar Machine Architecture

The MasPar Computer Corporation MP-2 is also massively parallel Single-Instruction, Multiple-Datastream (SIMD) computer, in this case consisting of from 1K to 16K processors[2, 10]. The architecture consists of a workstation front end, an array control unit, and the array of processors (Figure 1).

The MP-2 Array Control Unit (ACU) is a 12 MIPS scalar processor with a RISC-style instruction set and a demand-paged instruction memory. The ACU fetches and decodes MP-2 instructions, computes addresses

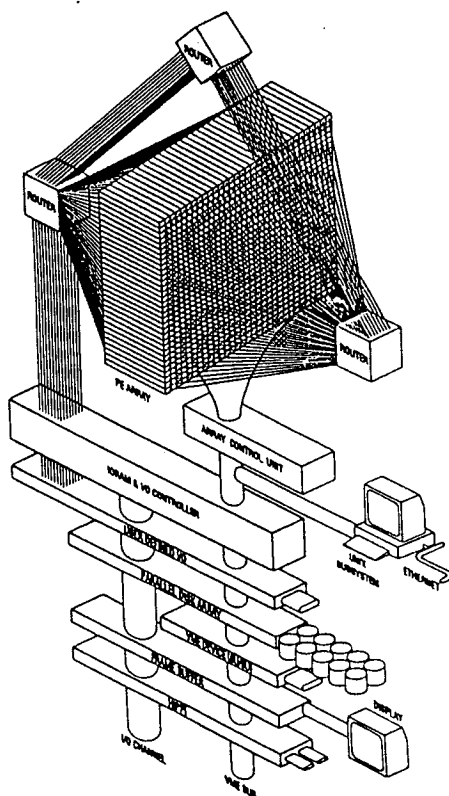


Figure 1: *The MasPar MP-2 system architecture.*

and scalar data values, issues control signals to the processor array, and monitors the status of the processor array.

Processing elements are custom RISC style 32-bit processors using a basic load/store style instruction set design and are clocked at 80 nsec. Each processor chip contains 32 of these individual processors. There is 64 Kbytes of off chip memory for each processor. Significant enhancements over the CM-2 processing element include hardware support for memory and computation overlap, and a more balanced memory bandwidth. Aggregate peak floating point performance for a 16K processor machine is 6.4 GFLOPS in single precision.

The architecture allows interprocessor communication to proceed in two ways. To perform random, or long distance communication, a three stage crossbar, circuit switched router may be employed. This general purpose network provides a peak bandwidth in excess of 1 GigaByte/sec in a 16K processor system. For more regular communication patterns, the machine may be viewed as a two dimensional grid with each processor having an 8 way connection to its neighbors with toroidal wraparound. This second network, termed X-Net, provides an aggregate bandwidth exceeding 20 GigaBytes/sec in a 16K processor system. The grid communication performance of the MP-1/MP-2 has been thoroughly investigated in [6].

Particle Simulation

Sequential Algorithm

For a small discrete time step, the molecular motion and collision terms of the Boltzmann equation may be decoupled. This allows the simulated particle flow to be considered in terms of two consecutive but distinct events in one time step, specifically there is a collisionless motion of all particles followed by a motionless collision of those pairs of particles which have been identified as colliding partners. The collisionless motion of particles is strictly deterministic and reversible. However, the collision of particles is treated on a probabilistic basis. The particles move through a cartesian grid of cells which serves to define the geometry, to identify colliding partners, and to sample the macroscopic quantities used to generate a solution.

The state of the system is updated on a per time step basis. A single time step can be thought of as comprised of five events:

1. Collisionless motion of particles.
2. Enforcement of boundary conditions.
3. Pairing of collision partners.
4. Collision of selected collision partners.
5. Sampling for macroscopic flow quantities.

Detailed description of these algorithms may be found in [9] and [3].

Data Objects Decomposition Approach

The data objects decomposition assigns one virtual processor to each particle in the simulation. This approach has been investigated in [4, 5, 11]. Figure 2 illustrates this decomposition. It has the advantage that the calculation dynamically adapts itself to the simulation such that there is good macroscopic load balance. Because each physical processor has been divided into an identical number of virtual processors, it follows that each physical processor is assigned an identical number of particles and therefore macroscopically has the same amount of work.

One of the more time consuming steps in a data-parallel implementation of a particle simulation is that of a global sort required on every time step. Sorting is necessary in order to determine which particles occupy the same cell and implicitly appears in the sequential algorithm as step 3. By sorting the particles, one obtains an arrangement as shown in figure 2, that is, particles occupying the same cell in physical space are represented by neighboring virtual processors in computational space. When the particles are ordered in this manner, cell boundaries may be quickly determined by having every processor compare its particle's cell index with that of its neighboring processor. Cell boundaries are often used to delimit segments in scan operations (to count number of particles in each cell for example). This convenient ordering is lost when

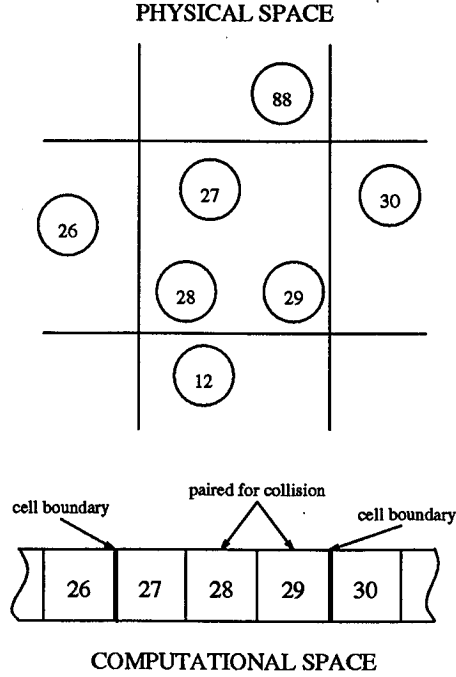


Figure 2: *Mapping of particle data to virtual processors in a data parallel implementation of a particle simulation. Each particle in physical space is represented by one virtual processor in computational space.*

particles move into different cells and the order must be restored by sorting. With the particle data sorted it becomes a simple matter to match pairs for collision as well as to sample for macroscopic flow quantities. The data parallel algorithms used to implement the five events comprising a time step are described in detail in [4, 5].

Discussion and Performance Results

The two codes were set up to run a channel flow simulation in a grid of $128 \times 8 \times 8$ cubic cells. The gas conditions were set for a Mach 5 flow and Knudsen number 0.375 based on the channel width. The simulation was started with 1,000,000 particles and was followed through 1,000 transient time steps followed by an additional 500 steps to collect an adequate number of samples to provide macroscopic results. Both codes have been individually validated.

All the CM-2 timings were for 8K processors of the CM-2 in the Numerical Aerodynamic Simulation (NAS) facility at NASA Ames Research Center. All MP-2 timings were obtained from an 8K processor machine located at MasPar Computer Corporation, thus a processor to processor performance comparison with the CM-2 is possible. All times are “elapsed” times on a lightly loaded machines. Although timings using greater numbers of processors are not presented, experience has indicated that the performance of the simulation essentially scales linearly with the number of processors when the problem size is scaled in a similar manner.

Performance for a particle simulation has been typically measured in terms of the average time required

Machine	Processors	Push Time (μ sec)
CM-2	8K	4.0
MP-2	8K	1.6
YMP	1	0.9

Table 1: *Performance for particle simulation on three systems.*

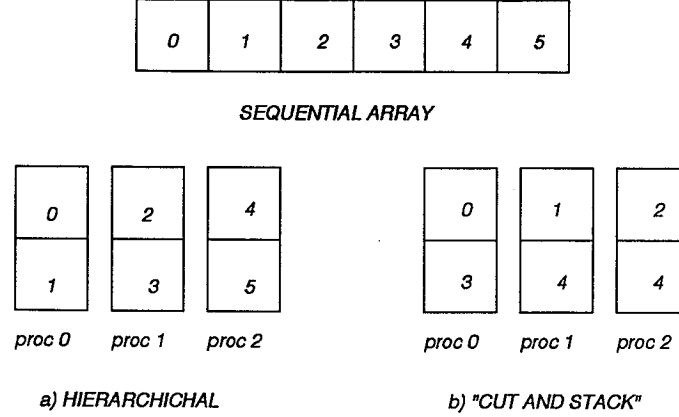


Figure 3: *Hierarchical versus "Cut and Stack" data layouts.*

to update a single particle during one time step (the "push" time in table 1). Table 1 presents performance using this metric for the two target machines and for comparison also presents the performance of a single processor Cray YMP on a fully vectorized implementation of the method. These measures are for a typical sampling step (i.e., they include the time for sampling).

Clearly, the MP-2 is exhibiting raw performance 2.5 times that of the CM-2. In this memory and communication intensive application, this superior performance is directly related to the more balanced memory and communication system inherent to the MP-2. This result is especially interesting in view of the radically different internal data layouts employed by the system software in the two machines. The CM-2 microcode employs a *hierarchical* layout whereas MasPar Fortran employs a *cut and stack* layout. The two layouts are illustrated in figure 3.

The main advantage of the hierarchical layout is in the reduction of communication along grid axes. For example, in figure 3, to perform a circular shift of data one processor to the right will require a total of 3 words to be communicated with the hierarchical layout compared to 6 words in the cut and stack layout. This reduction in communication for the hierarchical layout results in very efficient implementations of scan operations. This is important to particle simulation as scan operations are the primary mechanism for communicating macroscopic information (i.e. cell density or temperature) to the particles.

The main disadvantage with the hierarchical layout is that it can lead to many idle processors in dynamic or irregular data structures. Again, considering figure 3, if each array element represents a particle and only the first 3 particles are active in the simulation, then with the hierarchical layout instructions must still be

executed twice by each processor, whereas with the cut and stack layout instructions will be executed just once by each processor. The simulation problem described in this paper was sized to minimize this effect in the CM-2. For real problems it is not always possible to adjust the problem size to make maximum use of the machine, and in such cases the CM-2 performance suffers correspondingly.

It is interesting to note the differences in how the two implementations are spending their time during execution. Table 2 presents a profile, both as average microseconds per particle per step, and as percentage of total execution time, for each implementation on each of the components comprising a sampling time step. For the standing shock simulation, there is not a body in the flow and both implementations spend only a very small fraction of time in this part of the calculation. For this reason the boundary condition times are included in the motion component.

The sorting procedure dominates the calculation for both architectures. In table 2 the sorting procedure is broken into two parts, the ranking, which is determining the rank or position in the processor array where a particle's data must be sent, and the reordering, which is the actual communication of the particle data to the ordered position. On the CM-2 these two components amount to roughly half the computational cost. The ranking makes use of the `CMF_order` procedure in the CM Fortran Utility Library. This is a highly tuned, microcoded radix sorting routine which has excellent performance. For this reason, reordering the CM-2 takes longer than ranking. For this model problem, reordering amounts to eight permutations of about one million words each. The effective bandwidth for the CM-2 in this part of the calculation is thus 29 MB/sec.

The profile on the MP-2 is very similar to that for the CM-2, with the ranking taking a slightly larger percentage of the time. The ranking is written in the high level MPL language and cannot be considered fully optimized. Currently, ranking on the MP-2 is carried out with a bucket sort. Although the implementation depends primarily on the router, it has proved to be the most efficient small integer sort on the MP-2 with better performance than either Batcher's bitonic sort or a radix sort. This surprising outcome is attributable to the user control of virtualization on the MP-2, and is not directly dependent on the data layout. Consider that with user control of the virtualization and a hierarchical layout, one could allow each layer of the virtualization to independently increment the bucket counts and expect only a few collisions since particles in neighboring physical processors are likely to be in different cells. With the cut and stack layout, collisions can still be reduced by striding through the layers such that neighboring processors are not accessing the same memory layer while incrementing (and later decrementing) the bucket counts. This does incur a slight performance penalty since it requires local indirect addressing, however this is a small cost compared to the routing cost. Note that collisions are resolved arbitrarily, so it is not a stable sort; however stable sorting is not required for this problem.

Finally, for the reordering, the MP-2 achieves an effective bandwidth of 103 MB/sec. The loss of a factor of 5 over the peak bandwidth is attributable to injection time into the router and those collisions that occur despite striding through the memory layers to reduce collisions. Nonetheless, the measured bandwidth of the

Procedure	CM-2		MP-2	
	(μ s)	(%)	(μ s)	(%)
Collisionless motion of particles	0.40	10.1	0.06	3.9
Ranking	0.80	20.1	0.46	28.6
Reordering	1.12	28.1	0.35	21.5
Collision of selected collision partners	0.70	17.5	0.30	19.1
Sampling for macroscopic flow quantities	0.96	24.1	0.43	26.9

Table 2: *Profile for particle simulation on the two systems.*

MP-2 is about 3.6 times greater than that of the CM-2. The better application performance on the MP-2 compared to the CM-2 is largely attributable to the faster router.

Conclusions

The MasPar MP-2 provides better performance over the Connection Machine CM-2 in the particle simulation presented in this study. This advantage can be attributed to the faster router and improved memory performance of the MP-2. Memory and communication bandwidth requirements are especially important in this application, and these will have the greatest impact on the application performance. For this reason, adopting the hierarchical data layout employed by the CM Fortran compiler would conceivably further improve the application performance on the MP-2.

References

- [1] Bird, G.A., *Molecular Gas Dynamics*, Clarendon Press, Oxford, (1976).
- [2] Blank, T., "The MasPar MP-1 Architecture", *Proceedings of COMPCON Spring 90 - The 35th IEEE Computer Society International Conference*, San Francisco, CA, 1990.
- [3] Dagum, L., *On the Suitability of the Connection Machine for Direct Particle Simulation* PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford CA 94305, June 1990.
- [4] Dagum, L., "Three Dimensional Direct Particle Simulation on the Connection Machine", *AIAA 91-1365*, 1991.
- [5] Dagum, L., "Data Parallel Sorting for Particle Simulation", *Concurrency: Practice and Experience*, pp. 241-255, 4,3, 1992.
- [6] Fatoohi, R., "Performance Analysis of Four SIMD Machines", *RNR technical report 92-xxx*, NAS Applied Research Branch, NASA Ames Research Center, Moffett Field, CA 94035, 1992.
- [7] Levit, C., "Grid Communication on the Connection Machine: Analysis, Performance, Improvements," In H. D. Simon, editor, *Scientific Applications of the Connection Machine*, pp. 316-332. World Scientific, 1989.

- [8] Lumpkin, F.E., *Development and Evaluation of Continuum Models for Translational-Rotational Nonequilibrium*. PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford CA 94305, April 1990.
- [9] McDonald, J.D., *A Computationally Efficient Particle Simulation Method Suited to Vector Computer Architectures*, PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford CA 94305, December 1989.
- [10] Nickolls, J.R., "The Design of the MasPar MP-1: A Cost Effective Massively Parallel Computer", *Proceedings of COMPCON Spring 90 - The 35th IEEE Computer Society International Conference*, San Francisco, CA, 1990.
- [11] Wong, B.C., and Long, L.N., "Direct Simulation Monte Carlo (DSMC) on the Connection Machine", *AIAA 92-0564*, 1992.